MediaFileRenamer

A cross-platform tool for renaming media files

Alexander Nicklin
Candidate number: 2444

An AQA A-Level NEA Project (7517)

The Henley College, Henley-on-Thames, Oxfordshire
Centre Number: 62441

19th October 2025

Contents

1 Introduction and Declaration

1.1 Introduction

This goal of this project is the write a **utility program** for renaming files. The main target will be media files on the Ubuntu operating system.
Most Windows users will use the Bulk Rename Utility[4] to rename large numbers of files. BRU is an excellent and powerful tool but it has a few problems and weaknesses. They are, in order of importance:

- Not available on Linux
- Limited options to change media file's meta-data
- Not intuitive to use. A powerful utility once you understand it, but not easy for most casual users
- Code is closed-source
- Code is not free for commercial use

The design will be as simple as possible. Adding unnecessary back-ends, such as a database, will not be done. If a well known and maintained library for a feature is commonly available, it will be used.

1.2 Declaration

All code - except those modules listed below - and this report is my own work.
No previous or related utility program has been reviewed before designing and coding this project.
No search for other NEA's close to this one has been done. All code and design is my solo attempt to best deliver this project and expand on the programming aspects of the course.
Code Modules credited to other people:

- spdlog[2] is used as a logging library

2 Analysis

This is the Analysis section

3 Documented Design

This is the Documented Design section

4 Technical Solution

MediaFileRenamer has been written in C++ using the QT 6[1] cross-platform UI library. I have created build IDE's using CLion under Windows 11 and Ubuntu Desktop 25.10 to code and test this tool.
The C++ tooling ecosystem is a fractal nightmare of unbridled chaos https://github.com/marzer/tomlplusplus
Windows installers and Ubuntu packages will be created for end users of this tool. As this is a programming report, how these installers and packages were created is not documented here.

4.1 Creating a Windows Development Environment

Follow these steps to create a Windows development environment:

- Install QT from https://www.qt.io/download-qt-installer-oss
- Install Git from https://git-scm.com/install/windows
- Install CLion from https://www.jetbrains.com/clion/download/?section=windows
- Update all CLion Plugins
- Clone the MediaFileRenamer repository from https://github.com/Thentoxd/MediaFileRenamer
- MinGW toolchain dialog should appear - Click Next
- In the profile, add the following CMAKE options:

  ```
  "-DCMAKE_PREFIX_PATH=C:\Qt\6.10.0\mingw_64\lib\cmake"
  ```

- Build files should now be created
- Now load up main.cpp and click Run. The project should compile and execute the program
- Under cmake-build-debug/logs will be the logfile generated from the program

4.2 Creating a Linux Development Environment

We are using the gcc compiler (version 14.2.0) on Ubuntu Desktop 25.10.

```
chmod a+x qt-online-installer-linux-x64-4.10.0.run
sudo ./qt-online-installer-linux-x64-4.10.0.run

tar -xvzf CLion-2025.2.3.tar.gz

export Qt6_DIR=/opt/Qt/6.9.3/gcc_64

sudo apt-get install build-essential libgl1-mesa-dev
sudo apt-get install libxkbcommon-x11-dev

sudo apt install texlive-full
```

4.3 Tested Deployment Platforms

4.4 Legal

All my source code is available under the GPL licence. GPL is a copyleft license that requires all derivative works to also be released under the GPL, ensuring they remain open source

spdlog[2] is used as a logging library. Code uses the MIT licence, so can be used in both open source and proprietry programs.

4.5 Command Line Parameters

We are using the third-party CLI11 library to process command line parameters.

4.6 Runtime Logging

spdlog[2] is used as a logging library. This has been setup to provide one stream of messages that get sent to the console and to a three rotating 5Mb log files. Each log sink (console and file) can be set to log all message at or above a certain threshold. The logging also produces source code and line information, eg:

```
C:\Users\steph\CLionProjects\MediaFileRenamer\cmake-build-debug\MediaFileRenamer.exe
[2025-10-23 11:15:19.732] [multi_sink] [trace] [main.cpp:36] This is a trace level message
[2025-10-23 11:15:19.732] [multi_sink] [debug] [main.cpp:37] This is a debug level message
[2025-10-23 11:15:19.732] [multi_sink] [info] [main.cpp:38] This is a info level message
[2025-10-23 11:15:19.733] [multi_sink] [warning] [main.cpp:39] This is a warning level message
[2025-10-23 11:15:19.733] [multi_sink] [error] [main.cpp:40] This is a error level message 1
[2025-10-23 11:15:19.733] [multi_sink] [critical] [main.cpp:41] This is a critical level message
```

4.7 File Metadata Processing

https://github.com/Exiv2/exiv2, https://exiv2.org/examples.htmlexample1

4.8 Config File Support

https://github.com/nlohmann/json - didn't use in the end
toml++ is licensed under the terms of the MIT license - see LICENSE.

4.9 UI Developement

4.10 Documention

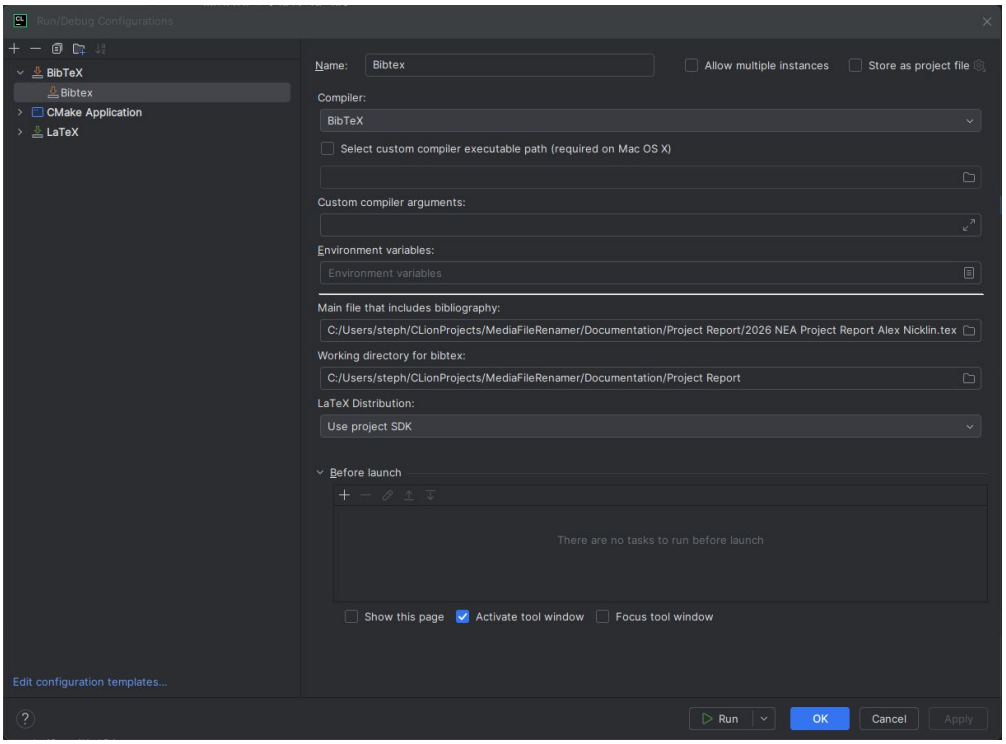
This Documentation was written in Latex. To edit and compile the Latex documentation, first:

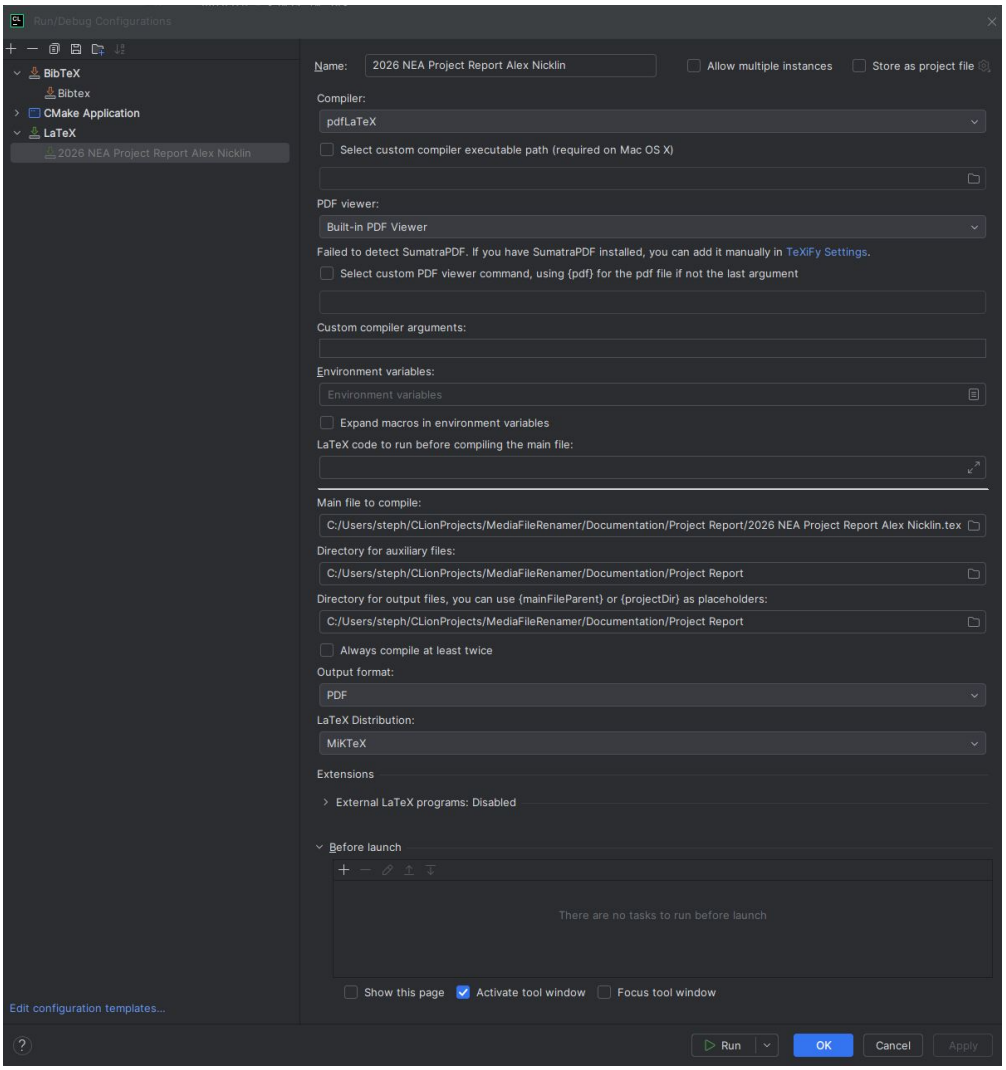
- Within CLion, install the extensions TeXiFy-IDEA and intellij-pdf-viewer
- Install MiKTeX from https://miktex.org/download
- Set MiKTeX to check and install updates automatically. Wait until it says updates are avaiable and do those
- I would recommend a full restart of the computer at this point

There are two source files for the documentation:

1. 2026 NEA Project Report Alex Nicklin.tex
2. 2026NEAProjectReportAlexNicklin.bib

[1] is the Latex source file for the report and [2] is the bibliography.
We used BibTeX and MikTeX to compile these two source files to make the final, single report file in pff format.
The Media subfolder contains any pictures referenced by [1].
In CLion, we need to add two custom Run/Debug configurations to "compile" [1] and [2].


Run/Debug Configurations
BibTeX
Bibtex
CMake Application
LaTeX
Name:
Bibtex
Allow multiple instances
Store as project file
Compiler:
BibTeX
Select custom compiler executable path (required on Mac OS X)
Custom compiler arguments:
Environment variables:
Environment variables
Main file that includes bibliography:
C:/Users/steph/CLionProjects/MediaFileRenamer/Documentation/Project Report/2026 NEA Project Report Alex Nicklin.tex
Working directory for bibtex:
C:/Users/steph/CLionProjects/MediaFileRenamer/Documentation/Project Report
LaTeX Distribution:
Use project SDK
Before launch
There are no tasks to run before launch
Show this page
Activate tool window
Focus tool window
Edit configuration templates...
Run
OK
Cancel
Apply



To fully rebuild the documentation, you have to do four compiles in this order:

1. LaTeX compile
2. BibTeX compile
3. LaTeX compile
4. LaTeX compile

5 Testing

5.1 Unit Testing

Testing with Catch2[3]

5.2 Valgrind

One of the weakness of C and C++ is memory control. It's easy to create memory leaks and memory errors (malloc 10 bytes, write 20 etc) which can corrupt the stack and result in difficult-to-trace bugs. To try and fix this, we use the Valgrind library to trace memory usage.

6 Evaluation

This is the Evaluation section

References

[1] The QT Group. *Qt for Open Source Development*. 2025. URL: https://www.qt.io/ (visited on 10/24/2025).

[2] Gabi Melman. *The Free File Renaming Utility for Windows*. 2025. URL: https://github.com/gabime/spdlog (visited on 10/24/2025).

[3] Catch Org. *A modern, C++-native, test framework for unit-tests*. 2025. URL: https://github.com/catchorg/Catch2/tree/devel (visited on 10/24/2025).

[4] TGRMN Software. *The Free File Renaming Utility for Windows*. 2025. URL: https://www.bulkrenameutility.co.uk/ (visited on 10/24/2025).